VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2019 Jan Rabčan

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jan Rabčan**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: MonkeyData, s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **doc. Mgr. Jiří Dvorský, Ph.D.**

Konzultant bakalářské práce: Bc. Tomáš Widlák

Datum zadání: 01.09.2018
Datum odevzdání: 30.04.2019

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 16. dubna 2019 .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 16. dubna 2019

MONKEYDATA
MonkeyData s.r.o.
Hladnovská 1255/28, 710 00 Ostrava
IČ: 02731452 DIČ: CZ02731452
www.monkeydata.cz

ING. JAN LAŠTŮVKA

Rád bych poděkoval společnosti MonkeyData s.r.o za to, že mi umožnila absolvovat bakalářskou práci ve formě individuální odborné praxe ve firmě. Dále děkuji za vřelé přijetí do týmu. Poděkování také hlavně patří konzultantovi Bc. Tomáši Widlákovi za cenné rady a také celému kolektivu za pomoc během vypracovávání odborné praxe. V neposlední řadě bych chtěl poděkovat také panu doc. Mgr. Jiřímu Dvorskému, Ph.D. za konzultace a odbornou pomoc při zpracování této bakalářské práce.

Abstrakt

Tato bakalářská práce popisuje proces autorova absolvování individuální odborné praxe ve společnosti MonkeyData s.r.o.. Firma se především zaměřuje na business eCommerce analytiku a vývoj analytických ecommerce nástrojů pro e-shopové platformy. Firma disponuje kvalitním technologickým programátorským zázemím. Hlavním cílem je seznámit čtenáře se zadáním praxe, její následnou analýzou a řešením. Náplní praxe bylo provést analýzu, navrhnout a implementovat webovou aplikaci, která bude zobrazovat statistické data, která jsou stažena z online e-shopových řešení zákazníka, kde jsou následně zpracována.

Klíčová slova: MonkeyData s.r.o, bakalářská práce, webová aplikace, AngularJS, MySQL, dxChart, analytický nástroj, graf, statistika, diplomová práce, LaTeX

Abstract

This bachelor thesis describes the process of the author's completion of an individual professional practice in MonkeyData s.r.o.. The company focuses mainly on eCommerce analytics business and the development of analytical ecommerce tools for e-shop platforms. The company has a high-quality technological programming background. The main aim is to acquaint the reader with the assignment of practice, its subsequent analysis and solution. The practice was to analyze, design and implement a web application that will display statistical data that is downloaded from the customer's online e-shop solutions where they are subsequently processed.

Key Words: MonkeyData s.r.o, bachelor thesis, web application, AngularJs, MySQL, dxChart, analytics tool, chart, stats, master thesis, LaTeX

Obsah

Seznam použitých zkratek a symbolů

API	–	Aplication programming interface
HTML	–	Hyper Text Metadata Language
OOP	–	Object oriented programming
Design pattern	–	Design of repeatable solutions
VIA	–	Vývoj internetových aplikací
VIS	–	Vývoj informačních systémů
PHP	–	Hypertext Preprocessor
Laravel	–	PHP framework
TypeScript	–	Object oriented JavaScript programming language
Grunt	–	JavaScript Task Runner
CSS	–	Cascading Style Sheets
LESS	–	Dynamic CSS
OOP	–	Object Oriented Programming
HTTP	–	HyperText Transfer Protocol
AngularJS	–	MVC JavaScript library
IDE	–	Integrated Development Environment
Jira	–	Issue & Project Tracking Software
Repozitář	–	Balíček obsahující kód
Cache	–	Uložení dat v paměti zařízení
Git	–	Source code version control tool
GraphApi	–	Graph - Aplication programming interface
Plugin	–	Aplikace, která se dá vložit do existující platformy
Back-end	–	Aplikace spuštěna na pozadí a nemá přímou interakci s uživatelem
Front-end	–	Část aplikace, kterou vidí uživatel
e2e	–	end-to-end testy aplikace

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Tato bakalářská práce vznikla jako završení posledního ročníku bakalářského studia vykonáním odborné praxe ve společnosti MonkeyData s.r.o.. Možnost pracovat v uvedené firmě na odborné praxi se naskytla po zveřejnění pozice frontend developera na projektu zmíněném v odborné praxi. V první části je představení společnosti MonkeyData s.r.o., jejich projektů a zázemí, do kterého jsem byl začleněn. Dále je zde popsáno mé zařazení do týmu a první dny zaškolování nebo také seznamování s prostředím a technologiemi, které budou provázet mou odbornou praxi až do konce. Ve společnosti MonkeyData s.r.o. jsem pracoval na projektu Inside, který je od kapitoly 5. popsán s jeho zadáním dílčích úkolů, jejich analýza a následná implementace. Poslední část je zaměřena na zhodnocení celého průběhu mé odborné praxe a vyzdvihnout zkušenosti nabyté během bakalářského studia na VŠB – Technické univerzitě Ostrava.

2 MonkeyData s.r.o.

2.1 O společnosti

Společnost MonkeyData s.r.o vznikla roku 2014 a od té doby se zabývá zpracováním dat z online e-shopových řešení zákazníků a následným doručením těchto obohacených dat např.: o predikci, historické statistiky, reprezentaci v grafech těmto uživatelům zjednodušující a šetřící čas.

Obrázek 1: Logo MonkeyData s.r.o.

Vedení, marketing, grafický a vývojářský tým sídlí v kancelářích podnikatelského a inovačního centra v Ostravě Vítkovicích. Zástupci společnosti se zúčastňují pravidelně mnoha tuzemských, ale i zahraničních konferencí zaměřené na eCommerce, BigData analytiky a analýzu dat.

2.2 Projekty

Společnost za svou existenci vyvinula několik, jak stand-alone online webových aplikací, tak i aplikací ve formě pluginů, implementovatelných do e-shopových řešení specifických platforem. V neposlední řadě firma také disponuje mobilními alternativami těchto webových aplikací, jak pro mobilní operační systém Android, tak i pro operační systém iOS. Tyto koncové aplikace by nefungovaly bez interních administračních, monitorovacích a importních aplikací, které jsou vytvořené na míru pro potřeby těchto koncových a řešení.

3 Proč jsem si tuto firmu vybral

Při výběru firmy, kde jsem měl absolvovat svou odbornou praxi, jsem zohledňoval několik hlavních kritérií. Jedna z nich byla důvěryhodnost firmy, její velikost a možnost vyzkoušet si technologie, které jsou například běžně placené, nebo také využití open-source projektů s odborným dohledem zkušených kolegů. Dalším kritériem bylo zaměření na vývoj webových aplikací, protože mě díky předmětu VIA zaujal tento okruh programování. Dále jsem taky očekával při úspěšném ukončení bakalářské praxe a bakalářského studia následnou spolupráci ve formě pracovní smlouvy.

Po přijetí požadavku na provedení bakalářské praxe konzultantem Bc. Tomášem Widlákem jsem následně nastoupil do firmy MonkeyData s.r.o. do mladého tříčlenného týmu pro vývoj front-end aplikací na pozici junior front-end developer.

3.1 Pracovní pozice

První den nástupu jsem musel projít krátkými přijímacími testy, které měly odhadnout můj aktuální stav a dovednosti potřebné k vypracování odborné praxe. Po přijímacím řízení mi byl vytvořen firemní email a následně vygenerovány přístupové údaje do všech interních systému, aplikací pro komunikaci a také organizaci a spravování úkolů. Během vykonávání odborné praxe mi byl velice nápomocný kolega Bc. Tomáš Widlák.

3.2 Prostředí a nástroje

Během dalších pár dní jsem měl za úkol nastudovat si Gitflow Workflow [6], který velice usnadňuje práci více lidí na jednom projektu zároveň. Ke komunikaci s každým oddělením je zde využita aplikace Slack [9], který je multiplatformní a umožňuje zasílat zprávy, přílohy a také uchovávat celou historii v případě placeného tarifu. K vytváření společných událostí pro porady, přednášky a dalších událostí týkající se organizace času se používá Google Calendar, který se dá nejen jednoduše použít v prohlížečích, ale také v mobilních zařízeních. Další nedílnou součástí mého vývojového zázemí bylo vývojové prostředí PhpStorm [7]. Disponuje velkou škálou interaktivních funkcí, jako našeptávání klíčových slov, zvýraznění těchto klíčových slov, varování výskytu programátorských nebo runtime chyb a mnoho dalších. K uchovávání kódu se používá webová nadstavba GitLab, která zapouzdřuje funkčnost Git [8] a ukládá kód do repozitářů.
K nahrávání kódu do repozitářů, spouštění testů a následné nahrání kódu na testovací/produkční prostředí je zde využita aplikace Jenkins [11].

4 O projektu Inside

Hlavním tématem mé praxe byl kompletní vývoj nové webové aplikace pro zákazníky e-shopových platforem, která by měla sloužit pro zobrazení eshopových dat s bohatší statistikou než nabízí tato eshopová řešení a základní mírou analytiky nad těmito daty. Dále by měla být možnost exportovat si vyfiltrovaná data do dále použitelného formátu jako je například csv[1] a nebo xslx[2]. Tento projekt zasahoval do všech základních sfér IT od databázového návrhu, přes importní systém, back-end řešení, API aplikační vrstvy a front-end aplikaci na které jsem se podílel během vykonávání této odborné praxe. Před hlavním vývojem si musel celý vývojový tým, vedoucí vývoje webových aplikací a vedení firmy sejít a důkladně zanalyzovat použitelné, jak databázové, tak i aplikační technologie s vysokou mírou škálovatelnosti jednotlivých aplikačních vrstev pro budoucí situace s velkou zátěží a rozložením výkonu na ně.

[1] csv - čárkami/středníky oddělené hodnoty, standardizovaný formát uložení dat
[2] xlsx - standardní formát. XML formát, kde jsou data organizovaná do buněk, spustitelné v Microsoft Excel

5 Aplikace Inside

Webová aplikace Inside, která slouží jako interaktivní analytický nástroj pro zobrazení/analýzu e-shopových dat zákazníka. Tuto aplikaci si každý zákazník specifické e-shopové platformy může aktivovat formou souhlasu stažení dat z e-shopové platformy a následně má možnost zobrazit si svá data v této webové aplikaci, která je základně embedovaná[3] v e-shopovém řešení, tudíž se jeví jako standardní stránka e-shopové platformy. Aplikace je rozdělena na dvě hlavní části podle použití.

5.1 Část první – Ověřený zákazník

Jakmile zákazník odsouhlasí stahování dat z jeho e-shopové platformy, může se dále vydat do webové aplikace, kde uvidí základní návod na použití aplikace ve formě tutoriálu. Zákazník se nemůže dále dostat, protože nemá stažená data potřebná pro zobrazení například pokročilé statistiky pro posledních 7, 30 nebo 360 dní. Po určité době, většinou 5-15 minutách je možné se dostat na dashboardy, kde uvidí základní statistiku. Dále zde má možnost vyfiltrovat si data pro konkrétní časové rozmezí, konkrétní stavy objednávek a následné exportování těchto dat. Dále je zde možnost změnit si výchozí měnu, ve které se budou stažené data v grafech zobrazovat.
V případě platícího zákazníka je zde i sekce nastavení, kde si zákazník může stáhnout výpis faktur nebo také dokupovat další placené doplňky aplikace. V systému má dále každý ověřený zákazník vytvořený projekt, podle kterého je identifikován například datový soubor s jeho online e-shop daty a nebo informace o jeho napojení.

5.2 Část druhá – Administrátor

V případě nějaké chyby nebo nesprávných dat[4], se mohou lidé zodpovědní za podpora a vývoj[5] připojit za tohoto uživatele a zobrazit poslední stránku, kterou zákazník viděl.

[3] embeded - schopnost zakotveného informačního systému v jiném informačním systému
[4] nesprávná data - data v dashboardech nejsou shodná se základními statistikami e-shopové platformy
[5] podpora a vývoj - zodpovědní zaměstnanci MonkeyData s.r.o.

6 Řešení úloh

V následující tabulce je odhadovaná časová náročnost v době analýzy projektu Inside v porovnání se stráveným časem zaznamenaným v průběhu vykonávání bakalářské praxe.

Úloha	Odhadovaný čas	Strávený čas
První	8 dnů	10 dnů
Druhá	7 dnů	8 dnů
Třetí	3 dny	3 dny
Čtvrtá	5 dnů	6 dnů
Pátá	4 dny	5 dnů
Šestá	4 dny	5 dnů
Sedmá	6 dnů	7 dnů

6.1 Úloha první – Analýza a Implementace grafu

Zadání úlohy

Můj první a nejobsáhlejší úkol, bylo společně s kolegou navrhnout a popsat základní chování obecného grafu, který bude dále rozšiřován na určité typy grafů. Obecný graf by měl obsahovat jednoduchou generičnost, která by měla obsahovat observables [10] na specifické události jako například: pro data, chybové situace a jiné stavy. Dále by měl disponovat jednoduchým mechanismem na zobrazení šablony stavů pro načítání dat, prázdné data a nebo data samotná.

Analýza úlohy

Před samotnou analýzou a implementací, jsem si musel sám nastudovat základy AngularJS frameworku [2] a také programovacího jazyka TypeScript [1]. Dále jsem mohl začít s vývojem obecného grafu. Vycházeli jsme z kolegových zkušeností z předchozí implementace podobných grafů, takže jsme nezačínali úplně od nuly, ale už jsme měli předběžný ověřený návrh obecného grafu a jeho vlastnosti jsme pouze rozšiřovali a zlepšovali.

Implementace

Jako první mi byly v Jira [12] aplikaci vytvořeny úkoly odpovídající jedné úloze popisující stručný obsah úkolu, jeho požadavky a grafický návrh. Mým úkolem je dále každou tuto úlohu posouvat v této aplikaci do specifického stavu jestli se na úkolu dělá/dělalo, nebo už je hotový, splněný ze strany programátora a připravený na Code Review. Jednotlivé třídy a části tohoto úkolu jsou psány v TypeScriptu a následně implementovány do AngularJS frameworku, kde jsou tyto části vykresleny. Pro většinu programování jsem použil programovací jazyk TypeScript.

Je to nadstavba nad jazykem JavaScript, který přináší do tohoto jazyka základy Objektově Orientovaného Programování a statické typování. Styly jsou psány pomocí Less [14]. Je to funkcionální rozšíření CSS [15]. K transkompilaci TypeScriptu na JavaScript zdrojový kód, čitelný pro prohlížeče a Less na CSS, také čitelný pro prohlížeče se používá webpack [5] nástroj. Ten umožňuje realtime provádění těchto operací, při změně konkrétních *.ts nebo *.less souborů. Webpack jsem našel jako alternativu za dosavadně používaný Grunt [13], který se používal složitěji a byl méně interaktivní. Jednotlivé komponenty grafu jako například: header, content, summary,
a další jsme implementovali do AngularJS aplikace formou komponent, jelikož AngularJS je component-based framework, což znamená, že v kódu je tato logika, struktura a styly pouze jednou a na stránce se vykreslují jen v případě že jsou žádané a mohou být do sebe tyto jednotlivé komponenty vnořené.

Základní stavební kámen je komponenta, kterou jsem pojmenoval `GraphBaseComponent`. Tato komponenta obsahuje základní vlastnosti a chování, které jsou pro všechny typy grafů stejné. Komponenty se obecně skládají ze tří hlavních částí a to jsou `template`, `controller` a `component`.

`template` - Je HTML reprezentace komponenty, AngularJS disponuje základními funkcemi, které obohacují HTML kód o logiku. Například: ng-if[6], ng-repeat[7], ng-show/ng-hide[8]

`controller` - Obsahuje logiku komponenty, definuje co se stane při načtení, vykreslení, zrušení komponenty. Dále také umožňuje přistupovat k proměnným v `template`.

`component config` - Specifikuje controller, template, bindings[9], transclude [16]

Jako první jsem musel naprogramovat funkcionalitu pro stahování dat. K tomuto se používá v AngularJS tzv. `Services`. Services slouží k distribuování dat mezi komponentami a nebo také například local storage[10]. Každý graf má své unikátní `GraphID`[11], tudíž je možné odkazovat na konkrétní endpoint[12] na GraphApi[13] odkud jsou specifická data vrácena pro konkrétní graf například v JSON formátu, což je jeden z nejrozšířenějších datových formátů pro výměnu dat. Jeho úspěch tkví převážně v čitelnosti a jednoduchosti. Dále existuje spousta prohlížečových rozšíření, které dokážou tento formát graficky lépe zobrazit, takže je práce s tímto formátem ještě jednodušší. Pro tuto funkčnost jsem využil design pattern observable, který jsem se naučil během bakalářského studia v předmětu VIS.

[6] ng-if [17] - if podmínka - provede se když je výsledek true
[7] ng-repeat [18] - jednoduchá smyčka, procházející všechny elementy v poli
[8] ng-show [19]/ng-hide [20] - při výsledku true se buď zobrazí/schová obsah
[9] bindings - vstupní atributy komponenty (řetězce, proměnné, funkce)
[10] local storage - interní paměť prohlížeče
[11] GraphID - identifikátor, který se shoduje s ID endpointem na GraphApi
[12] endpoint - url adresa, ze které je možné získat data
[13] GraphApi - API pro grafy

Hlavní výhodou tohodo návrhového vzoru je to, že mám jednu roli publishera a několik instancí subscriberů, kteří provedou akci, kdykoliv publisher změní obsah/data. Dále jsem musel navrhnout podle grafického návrhu obecnou HTML strukturu celého grafu a tímto tedy rozdělit celý graf do `header`, `content`, `loading`, `error` a `modal` komponenty. Dále jsem implementoval specifické chování v konkrétních stavech grafu například při načítání dat, se zobrazí `loading` komponenta a nic jiného není vidět. Při úspěšném stažení dat se zobrazí `content` komponenta a v případě neúspěšného stažení dat se zobrazí `error` komponenta zobrazující chybovou hlášku co se stalo a případně obnovovací tlačítko, díky kterému se pokusí graf stáhnout data znovu.

Nad rámec úkolu jsem implementoval i třídu `GraphManager`, který je tvořený návrhovým vzorem factory a prochází skrze něj každý inicializovaný graf na stránce. A následně umožňuje pracovat s konkrétními nebo všemi grafy z venku najednou například hromadným znovu získáním dat pro grafy. Dále každý graf disponuje funkcionalitou pro export, to jsem ale po implementaci odstranil z obecného grafu, protože každý typ grafu má specifický datový set, takže jsem v hodnotovém grafu nemohl exportovat data stejným způsobem jako v tabulkovém grafu. Dále jsem implementoval konfiguraci na možnost načítání při změně časového rozmezí nebo jestli se má graf načítat při prvním načtení stránky a grafu v něm.

Každý obecný graf má tedy toto základní rozhraní, které definuje uložiště dat pro každý jednotlivý graf a možnost přístupu k těmto datům. Každý graf disponuje proměnnou `id`, která identifikuje specifický graf a zároveň se tímto `id` odkazuje na GraphApi endpoint, odkud se získávají data. Dále je zde `graphType` typu enum, který může nabývat hodnot table, spline, ico a tím pádem si může každý jednotlivý graf vybrat předem nadefinovanou šablonu, pro konkrétní typ grafu. Proměnné `loadingObservable`, `dataObservable`, `errorObservable` pouze uchovávají poslední uloženou hodnotu. Každý graf má také veřejně dostupnou metodu `reload`, která umožňuje znovu načtení dat grafu.

```
1  export interface IGraph {
2      readonly id: number;
3      graphType: EnumGraphType;
4      loadingObservable: IObservable;
5      dataObservable: IObservable;
6      errorObservable: IObservable;
7
8      reload(): ng.IPromise<IGraphApiData>;
9
10     request: IGraphApiRequest;
11 }
```

Výpis 1: Obecné rozhraní pro všechny následující implementované grafy

6.2 Úloha druhá – Analýza a Implementace GraphApi

Zadání úlohy

Mým dalším úkolem bylo implementování základního RESTful API [21] endpointu pro získávání dat pro konkrétní grafy. Sestavení SQL dotazu a následné získání dat z databáze, provedení základních matematických operací nad těmito výslednými daty, které jsou dále odeslány konkrétnímu grafu. Předem víme, jaké grafy a jaký datový základ budou tyto grafy vyžadovat, takže se mohou všechny grafy v tomto úkolu implementovat.

Analýza úlohy

Jelikož nové řešení GraphAPI by bylo časově dost náročně, rozhodlo se, že se použije stávající aplikace GraphApi, takže mě jen zbývalo oprášit základní znalosti PHP, naučit se základy Laravel [3] frameworku a následně na pár grafech prostudovat aktuální možnosti projektu GraphApi a teprve poté implementovat endpointy pro nové grafy pro Inside projekt.

Implementace

Během základního návrhu a implementace komponent pro obecný graf, bylo důležité také navrhnout datovou strukturu ve které se bude graf doptávat se specifickými parametry na konkrétním endpointu aby dostal správná data. V tomto případě se muselo nadefinovat vstupní HTTP request atributy, které budou akceptovatelné pro konkrétní grafy. Každý graf se může lišit vstupními proměnnými, zdrojem dat, výpočtem nad těmito daty a také formátem výstupu. Pro můj prvotní graf abych si vyzkoušel vytvoření vlastního grafu mi stačilo implementovat výběr všech objednávek a jejich celkový příjem s/bez DPH. Pro uložená data se používá MariaDB [22] server, takže jsem zde využil základní znalosti ze studia z databázových předmětů pro pokročilejší tvorbu SQL dotazů.

Zde přichází na řadu první konfigurace grafu a to jsou vstupní proměnné. Mezi nejpoužívanější proměnné co přichází společně s dotazem z grafu jsou časové rozmezí ve kterém se má hledat a pro jaký projekt se mají data stahovat. Každý HTTP request grafu obsahuje čístlo projektu v zašifrované podobě, takže tento atribut nelze zfalšovat, nebo získat data jiného projektu.

Dále je zde možná konfigurace na úrovni měny. V jiném projektu se stahují pravidelně, denně aktuální kurzy nejpoužívanějších měn, na které je možné poté převádět z historicky stažených dat pro objednávky nebo produktů.

Další je nejpoužívanější konfigurace filtrování podle stavů objednávek. Každá e-shopová platforma disponuje určitou sadou stavů objednávek, které dále můžeme využít ke shlukování dat, případně filtrování podle stavů objednávek, které chceme vidět. Například si můžeme vyfiltrovat objednávky, které jsou zaplacené a doručené, tudíž nebudu mít ve statistikách nedoručené nebo zrušené objednávky.

Implementace každého unikátního grafu spočívá v jeho definici SQL, díky kterému získáme datový základ, na kterém dále můžeme použít agregační funkce jako třeba shlukování podle měsíce, dnů a nebo průměrování konkrétních hodnot a dalších.

Výsledný SQL dotaz mého prvnotního grafu, vypadá například takto. Jako první jsem si vybral sloupce `date_id`, `price`, `price_without_vat` v select sekci a přidal jim čitelné aliasy.

Dále jsem specifikoval název tabulky, ze které se vybírají data a dále jsem specifikoval pro jaký projekt chci zobrazit data, jaké stavy objednávek chci vyfiltrovat a v jakém časovém rozmezí chci data vrátit.

```
1 select `base`.`date_id` as `date`,
2        `base`.`price_without_vat` as `revenue_without_vat`,
3        `base`.`price` as `revenue_with_vat`
4 from `f_eshop_order_XXX` as `base`
5 where `base`.`project_id` = xxx
6   and `base`.`order_status_id` in (xxx, xxx, xxx)
7   and `base`.`date_id` >= 20180101
8   and `base`.`date_id` <= 20181231
```

Výpis 2: Základní SQL dotaz pro získání objednávek a jejich revenue

Dále se tento datový výstup vloží do PHP pole a je s ním dále pracováno v případě agregačních funkcí se provede například shlukování a přepočítávání dat podle měsíců/dnů/dostupných kategorií a jako poslední se zde aplikují formáty na měny nastavené pro daný projekt, nebo podle atributu v requestu.

Po seznámení s funkčnostmi GraphApi a specifikací konfigurace grafů jsem mohl začít implementovat konkrétní grafy a jejich nastavení na které bude každý graf reagovat. Většinu logiky jsem nemusel implementovat, protože již naimplementována byla ve stávající aplikaci, jen jsem převzal tuto konfiguraci z jiných grafů s podobným datovým základem a implementoval do nově vytvořených grafů.

Konfiguraci, kterou jsem si musel sám implementovat byla specifikace dimenzí[14] a metrik[15] pro jednotlivé grafy. Znamenalo to, že jednotlivý endpoint pro graf mohl přijímat různý počet dimenzí a metrik podle kterých se shlukovaly data do sebe již při SQL dotazu nebo v agregačních funkcích nad hotovými daty.

Další nedílnou součástí, na kterou se nesmělo zapomenout bylo navrhnutí cachování těchto dat, pro urychlení načítání dat. Je to způsob kdy se výsledné data ukládají do paměti serveru, která je mnohem rychlejší než rychlost disku. Díky tomuto zadání jsem měl za úkol seznámit se s technologií redis, která funguje na bázi ukládání/čtení dat z/do paměti.

[14] Dimenze - slovníkové, ne-číselná data - slouží ke shlukování výsledků
[15] Metriky - číselná data - reprezentace hodnoty dimenze

6.3 Úloha třetí – Implementace hodnotového grafu

Zadání úlohy

Jako první naprogramování šablony grafu byl hodnotový graf. Graf disponuje jednou až dvěma agregovanými hodnotami, tudíž je jeho implementace jednoduchá. Dále je zde možnost vyexportovat data konkrétního grafu do csv nebo xlsx formátu. V neposlední řadě by měl tento obecný graf obsahovat testy.

Obrázek 2: Příklad - hodotové grafy

Analýza úlohy

Před úkolem bylo potřeba navrhnout všechny vstupní konfigurace, které jsou při získávání dat z GraphAPI zasílány společně s HTTP requestem na tyto data.

Implementace

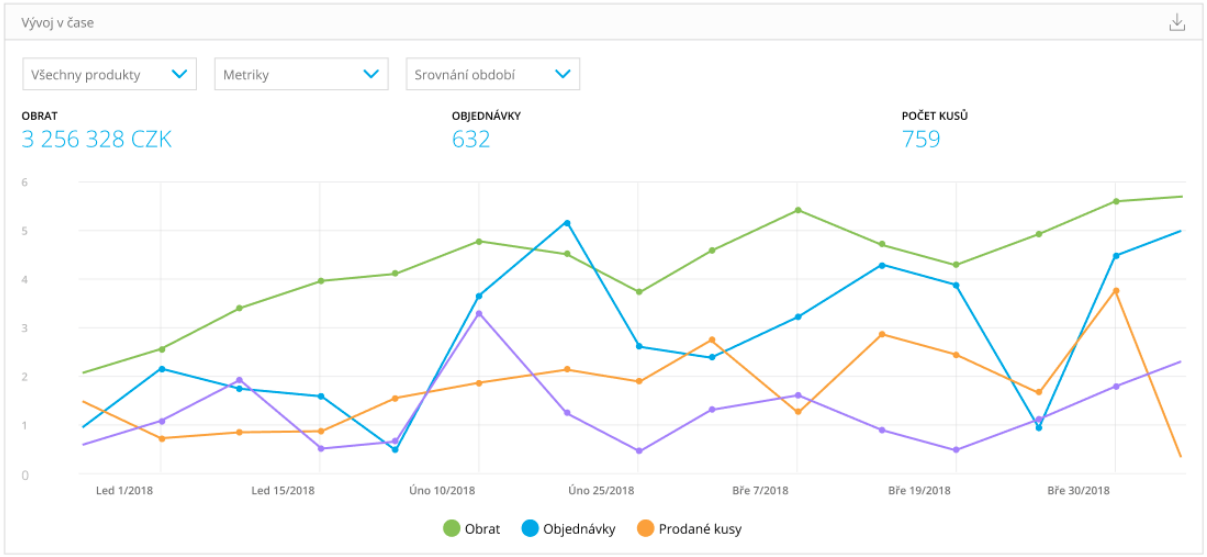
Po zhotovení základní podoby obecného grafu a definice endpointu na GraphAPI jsem mohl začít rožšiřovat komponentu obecného grafu o strukturu hodnotového grafu s názvem `GraphIcoComponent`, ta následně dědí z `GraphBaseComponent`. Tato komponenta ještě není finální případ specifického grafu, slouží pouze jako rozšíření pro všechny hodnotové grafy. V případě hodnotových grafů GraphAPI vrací v HTTP response všechna data pro celé časové rozmezí a také agregované hodnoty v label objektech, ve kterých je specifikace jestli je číslo měna/procento/číslo, podle kterého se zobrazují dále tyto hodnoty v grafu. Data v časovém rozmezí je zde potřeba v případě, že bude chtít zákazník exportovat data u těchto grafů. Dále většina hodnotových grafů obsahuje tooltip pro nápovědu co obsahuje konkrétní graf případně i výpočet.

Dále jsem si zde vyzkoušel jak se provádí end-2-end [23] testování webových aplikací. Převážně se toto testování používá pro ověření, jestli se webová aplikace chová od začátku do konce jak má, podle specifického scénáře. Tyto testy mohou být spustitelné při vývoji na počítači, protože nejsou náročné na výkon a zároveň by měly být úspěšné před operaci push do GIT repozitáře. V případě chyby by se neměl nahrát kód na produkční/testovací prostředí.

6.4 Úloha čtvrtá – Implementace liniového grafu

Zadání úlohy

Další komplexnější typ grafu, který jsem měl za úkol implementovat byl liniový graf za využití dxChart[16] knihovny. Liniový graf disponuje konfigurací vyfiltrování dimenzí, metrik a nebo funkcí na porovnání aktuálního vyfiltrovaného období s obdobím předchozím. Dále je zde možnost zobrazené data exportovat do csv nebo xlsx.



Obrázek 3: Příklad - liniový graf

Analýza úlohy

Prerekvizitou tohoto úkolu mi bylo nastudování základního použití dxChart knihovny, konkrétně liniového grafu a potřebnou konfiguraci, jak se zobrazují data v tomto typu grafu.

Implementace

Jako první věc jsem našel odkaz na GitHub repozitář dxChart projektu a nadefinoval jsem řádek do `package.json` soboru, kde se definují stahující se balíčky třetích stran. Poté jsem mohl dále pracovat s touto knihovnou. Pro tento typ grafu jsem vytvořil `GraphSplineComponent` komponentu, ta následně dědí z `GraphBaseComponent` a sloužila jako šablona pro linové grafy. Dále jsem implementoval základní konfiguraci pro liniové grafy na zobrazení filtrů pro dimenze, metriky a porovnání období. Stažená data nejsou v tomto případě rovnou zobrazena v grafu, ale jsou předána do dxChart liniového grafu, který data vykreslí v svg elementu. Knihovna obsahuje spoustu funkcí nad tímto typem grafu, ale potřebnou možnost schování/zobrazení linky v grafu

[16] dxChart - JasaScript knihovna, obsahující většinu základních grafů

po kliknutí na konkrétní legendu jsem musel implementovat sám za mírného experimentování. Dále disponuje konfigurací pro tooltipy nad jednotlivými hodnotami liniového grafu. Jelikož graf získává data pro tento graf v časových rozmezích typu den, týden a měsíc, tak jsem musel implementovat jednotlivé formátování pro každé rozmezí.

Časové hodnoty se vrací z GraphApi ve formě unix time [24], který je v UTC časové zóně a může být následně formátován podle klientského prohlížeče. Na časovém rozmezí také závisí zobrazení vertikálních os a zobrazení horizontálních hodnot v samotném liniovém grafu. Proto se musí tato konfigurace při každém získání nových dat z GraphApi nastavovat opět znovu.

Podle příchozích dat se musela aktualizovat také legenda, které umožňuje schovat/zobrazit jednotlivé čáry reprezentující hodnoty metrik v časovém rozsahu.

Každý liniový graf má dále proměnnou `transclude`, která umožňuje vložit komponentu nebo HTML kód do této komponenty. Při změně dat pro tento liniový graf se zavolá metoda `change` a tím se dá vědět rodičovské komponentě, že byly znovu získáná data a v tom případě může třeba odeslat nově získaná data do jiné komponenty.

```
1 export default class GraphSplineComponent implements IGraphComponent {
2     controller = GraphSplineComponentController;
3     template = graphSplineTemplate;
4     bindings = {
5         setting: '<',
6         change: '&?',
7     };
8     transclude = true;
9 }
```

Výpis 3: Konfigurace komponenty liniového grafu

6.5 Úloha pátá – Implementace tabulkového zobrazení

Zadání úlohy

Jako poslední typ grafu, který jsem měl implementovat byl tabulkové zobrazení. Datově se moc neliší od liniového grafu, pouze zobrazuje data v tabulkové podobě. Graf by měl mít možnost filtrování stažených dat, sumarizaci jednotlivých sloupců a virtuální scroll při velkém objemu dat.

Produkty

Metriky

	Název produktu	Obrat vč. DPH	Objednávky	Počet kusů
	Mobily			
1.	Samsung Galaxy S9 Duos	286 785 CZK	13	13
2.	iPhone 8 64GB	271 951 CZK	13	10
3.	Samsung Galaxy A5 (2017)	150 873 CZK	10	17
4.	iPhone 7 32GB	135 912 CZK	8	8
	Celkem: 4	Celkem: 3 256 328 CZK	Celkem: 632	Celkem: 759

Obrázek 4: Příklad - tabulkové zobrazení

Analýza úlohy

Z důvodu časového tlaku a komplexnosti požadavků úkolu jsme využili na tento typ grafu opět dxChart knihovnu, která většinu kritérii pro tento úkol splňuje rovnou, bez složité konfigurace a potřeby hlubokých znalostí.

Implementace

Pro šablonu tohoto grafu jsem implementoval podobnou strukturu jako u liniového grafu. Výslednou komponentu jsem pojmenoval `GraphTableComponent`, ta následně dědí z `GraphBaseComponent`. Pro tento typ grafu narozdíl od ostatních jsem musel implementovat jinou exportní funkci, protože dxChart nenabízí standardně vyfiltrované všechny řádky, ale pouze ty co už jsou vykreslené na stránce, tudíž jsem musel implementovat vlastní filtr nad staženými daty, který se musel shodovat s tím co má standardně tato knihovna zakomponované v sobě. Dále jsem zde musel přestylovat celou tabulku pomocí LESS, protože vzhled, který je defaultní od dxChart knihovny neladil s grafickým návrhem.

Každá tabulka vychází z této konfigurace. Od liniového grafu se liší tím, že má nadefinované navíc dvě metody `onInitialized` a `onContentReady`. Metoda `onInitialized` se zavolá při prvním vykreslení tabulkového zobrazení na stránce. V této situaci mohu schovat specifický loading, který je specifický pro graf, který se načetl poprvé na stránce. Metoda `onContentReady` se zato volá při každém novém získání dat pro tabulkové rozhraní. V tuto chvíli mohu volat jednoduchý loading, který nezakryje celý graf, ale jen vnitřní část grafu, kde jsou zobrazena data.

Na rozdíl od liniového grafu nebo hodnotového grafu, musí mít tabulkové zobrazení předem definovanou konfiguraci obsahu ve formě sloupců a jejich nastavení, kde je předem určený datový typ pro formátování buněk, název sloupce, podle kterého se zobrazí data do buňky pro konkrétní řádek, jeho šířka, výška a název sloupce viditelný v tabulce.

```
1
2  export default class GraphTableComponent implements IGraphComponent {
3      controller = GraphTableComponentController;
4      template = graphTableTemplate;
5      bindings = {
6          setting: '<',
7          change: '&?',
8          onInitialized: '&?',
9          onContentReady: '&?'
10     };
11     transclude = true;
12 }
```

Výpis 4: Konfigurace komponenty tabulkového zobrazení

6.6 Úloha šestá – Dashboard – Souhrný přehled

Zadání úlohy

Na stránce budou základní grafy obsahující analytiku na úrovni objednávky. Podle grafického návrhu implementujte a dodržte rozložení grafů.

Analýza úlohy

Získal jsem grafickou analýzu dashboardu pro souhrný přehled a specifikace pro všechny grafy na této stránce.

Implementace

Jako předposlední úkol a tudíž jeden z finálních se kterým se setkají zákazníci bylo implementovat předem specifikované grafy na konkrétním dashboardu. Dostal jsem grafický návrh a dle něj jsem následně implementoval graf po grafu na specifická místa s konfiguracemi. Mezi prvními byly grafy hodnotové pro jednoduchou přehlednost a shrnutí základních metrik v časovém rozmezí pro zákazníka. Patří mezi ně `počet objednávek`, `celkový obrat`, `průměrná hodnota objednávky` a `prodané kusy`.

Specifická konfigurace a postup pro `počet objednávek` je následující.

1. Vytvořil jsem komponentu s názvem `Graph466Component` a ta následně dědila z `GraphIcoComponent`, takže jsem měl připravený hodnotový graf.
2. Specifikuji konfiguraci prvotního načítání nebo načítání při změně časového rozmezí pro tento konkrétní graf podle potřeby.

Jelikož není v hodnotových grafech jakákoliv jiná parametrizace, není potřeba další konfigurace na straně front-end aplikace.

Dále podle grafického návrhu se na stránce nachází dvě tabulkové grafy. První naimplementovaný tabulkový graf je seznam `produktů` a jejich obrat s počtem prodaných kusů. Druhý byl seznam `zákazníků` a jejich utraceným obratem a počtem objednávek, které provedli v časovém rozmezí.

Specifická konfigurace a postup pro tabulkový graf seznamu `produktů` je následující

1. Vytvořil jsem komponentu s názvem `Graph470Component` a ta následně dědila z `GraphTableComponent`, takže jsem měl připravený tabulkový graf podle potřeby.

2. Specifikuji konfiguraci prvotního načítání nebo načítání při změně časového rozmezí, výšky grafu pro tento konkrétní graf.

3. Dále ale také musím specifikovat nastavení, které se dále pošle do dxChart komponenty, kde se následně vykreslí.

4. A jako poslední jsem specikoval seznam dimenzí, což je v tomto případě pouze produkt, seznam metrik, což je v tomto případě obrat a počet prodaných kusů.

Dále jsem měl za úkol podle grafického návrhu naprogramovat/najít datepicker komponentu a select komponentu pro statusy objednávek. Díky tomuto úkolu jsem se více dostal do hloubky architektury komponent a sdílení dat mezi těmito komponentami. Dále se tyto komponenty využily v dalším dashboardu, kde sloužily jako filtrování pro grafy.

6.7 Úloha sedmá – Dashboard – Pokročilé statistiky

Zadání úlohy

Další a poslední úkol bylo implementovat dashboard pro pokročilé statistiky. Na stránce budou základní grafy obsahující analytiku na produktu. Dashboard obsahuje několik základních grafů, ale také taby obsahující předdefinované grafy a v neposlední řadě tabulkový graf PRO, umožňující zákazníkovi nakombinovat dimenze a metriky podle potřeby.

Analýza úlohy

Podle názvu je vidět, že se bude jednat o komplexní dashboard s mnoha funkcemi. Pro implementaci změny tabů, zobrazení grafů a současně změnění URL jsem si musel v dokumentaci AngularJS nastudovat jak se mění state[17]

Implementace

Podle grafického návrhu se zde opět vyskytují hodnotové grafy ale s lehce odlišnými metrikami. Patří mezi ně `prodané kusy`, `Tržby za produkty`, `slevy uplantněné na produkty` a `průměrná prodejní cena produktu s/bez DPH`. Co se týče hodnotových grafů, tak jejich implementace je skoro stejná jako u dashboardu pro souhrný přehled, akorát mají jiné ID, díky kterému dostanou z GraphAPI jiná data.

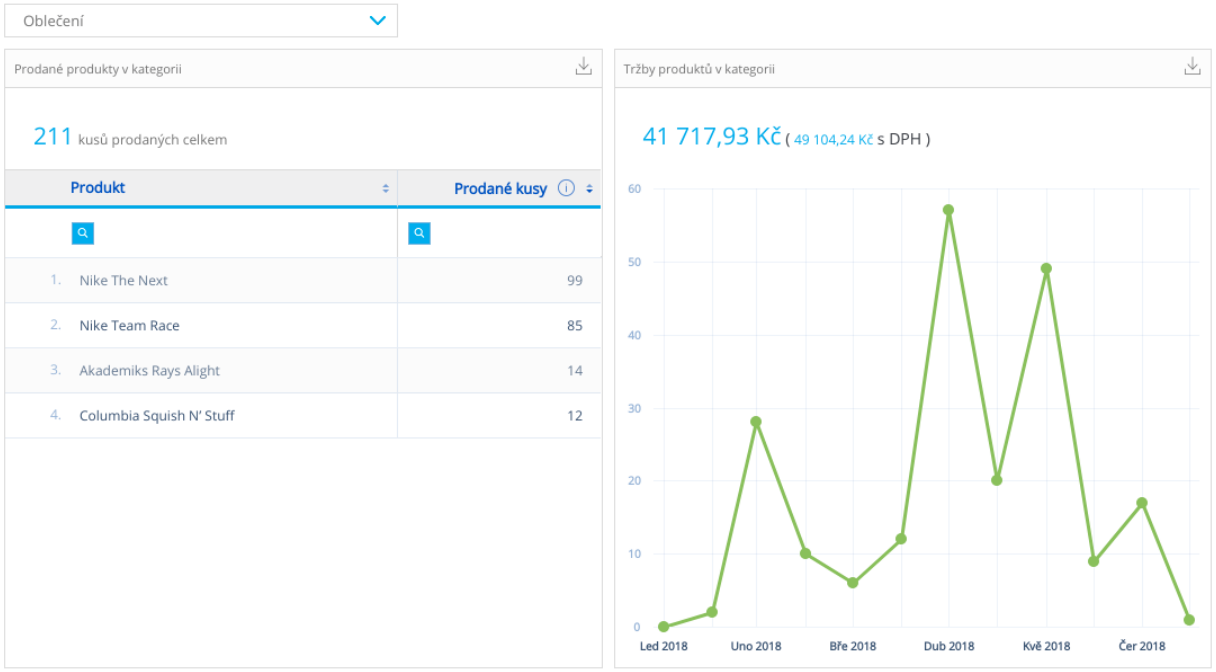
Změnou oproti souhrnému přehledu bylo, že jsem musel navrhnout a implementovat responzivní zobrazení grafů v tabech. To umožňuje zobrazit celý graf případně i více grafů na celou šířku prohlížeče s možností jednoduše mezi grafy přepínat.

Jako první jsem implementoval tabulkový graf pro produkty, u něj jsem musel povolit jen zobrazování metrik pro produkty. Další obsahoval seznam výrobců, který měl stejnou konfiguraci pro metriky jako graf s produkty. U dalšího tabu pro hity měsíce, propadáky měsíce a kombinace produktů jsem musel implementovat jiný způsob formátování, protože tyto sloupce obsahovaly více hodnot. Například v hitech měsíce obsahoval jeden sloupec hodnotu jak se zlepšil prodaný produkt za posledních 30 dní a pak i jeho předchozí hodnotu a aktuální hodnotu.

Předposlední tab byl PRO tab, který umožňoval zákazníkovi nastavit si libovolný počet dimenzí a metrik v tabulce a tím si agregovat data podle potřeb. Tahle tabulka byla pro mě nejkomplikovanější. Osobně jsem měl s touto tabulkou začít, abych měl jednoduchou implementaci ostatních tabulkových grafů, kde by se nadefinovaly jen potřebné metriky a dimenze, protože jsem do ní nadefinoval všechny možné zvolitelné dimenze a metriky, tudíž každá menší tabulka např. produkty obsahují dimenzi produkty a základní metriky a to je vše potřebné pro tabulkový graf. Což se, ale na začátku vývoje netušilo, ale vzal jsem si z toho ponaučení pro příští vývoj.

[17] state - reprezentace aktuální stránky

Poslední tab obsahoval dva grafy, takže jsem musel vytvořit komponentu `GraphCategoryComponent`, která obalovala graf pro prodané produkty v kategorii a tržby produktů v kategorii.



Obrázek 5: Pokročilé statistiky - Tab - grafy pro kategorie

Dále jsem zde musel implementoval select komponentu, která z GraphApi získává seznam kategorií, který je následně využitý v těchto grafech jako filtr pro data. V konfiguraci těchto dvou grafů jsem musel tedy nastavit, že se nemají načítat při prvotním načtení stránky ani při změně časového rozmezí, ale doimplementovat načtení při změně časového rozmezí.

6.8 Řešení daného úkolu v čase

V následující tabulce shrnu denní harmonogram během vykonávané odborné praxe ve firmě.

Úloha	Využitý počet dní
Seznámení s firemním prostředím	1 den
Instalace počítače, programů, přidělení SSH klíče	1 den
Specifikace zadání a seznámení s problematikou	2 dny
První úloha - Analýza a Implementace grafu	10 dnů
Druhá úloha - Analýza a Implementace GraphApi	8 dnů
Třetí úloha - Implementace hodnotového grafu	3 dny
Čtvrtá úloha - Implementace liniového grafu	6 dnů
Pátá úloha - Implementace tabulkového grafu	5 dnů
Šestá úloha - Dashboard - Souhrný přehled	5 dnů
Sedmá úloha - Dashboard - Pokročilé statistiky	7 dnů
Testování funkčnosti a ladění chyb	2 dny

7 Závěr

7.1 Získané a uplatněné znalosti

Praxe převážně vyžadovala znalosti TypeScript a znalosti kódování, které jsem si zdokonalil během vykonávání této odborné praxe. Dále jsem zde nabyl zkušenosti v oblasti databázových systémů. Mezi předměty, které mě připravily na programovací návyky byly převážně Algoritmy I a II či Programování I a II. Dále jsem zde nabyl zkušenosti a přístup k odhalování chyb v kódu za využití IDE prostředí. Nedílnou součástí bylo také využití znalostí z předmětu Programovací jazyky, které mě připravily na objektově orientované programování. Dále jsem zde uplnatnil vědomosti z Úvodu do softwarového inženýrství a Úvodu do databázových systémů získané během studia.

7.2 Chybějící znalosti

Díky všeobecnému základu, který mi během studia na vysoké škole poskytla, jsem během vykonávání odborné praxe neměl tolik nedostatku co se týče znalostí a dovedností. Jedním nedostatkem, který jsem neznal byla znalost verzovacího systému Git [8] a jeho usnadnění ve vývoji více lidí nad jedním úkolem.

7.3 Výsledky a celkové zhodnocení

Z mého pohledu byla praxe velice přínosná a jsem rád, že jsem zde mohl uplatnit zkušenosti získané během studia převážně z předmětu Vývoj internetových aplikací, Vývoj informačních systémů, díky kterému jsem byl schopen občas během analýzy oponovat co se týče návrhu a implementace a v neposlední řadě Programování. Dále jsem díky rozdělení úkolů, jsem mohl na jednotlivých částech pracovat sám a tím se učil samostatnosti co se týče programování. Nejvíc mě dále zaujala práce v týmu, sdílení informací a komunikace mezi sebou.

Literatura

[1] *Jazyk TypeScript* [online] [2019-04-23]
Dostupné z `https://www.typescriptlang.org/`

[2] *Dokumentace AngularJS Framework* [online] [2019-04-23]
Dostupné z `https://angularjs.org/`

[3] *Dokumentace Laravel Framework* [online] [2019-04-23]
Dostupné z `https://laravel.com/`

[4] *Dokumentace dxChart JS knihovna* [online] [2019-04-23]
Dostupné z `https://js.devexpress.com`

[5] *Dokumentace webpack nástroj* [online] [2019-04-23]
Dostupné z `https://webpack.js.org/`

[6] *Článek o Gitflow Workflow* [online] [2019-04-23]
Dostupné z `https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow`

[7] *Dokumentace IDE PhpStorm nástroj* [online] [2019-04-23]
Dostupné z `https://www.jetbrains.com/phpstorm`

[8] *Git - Distributed version control system* [online] [2019-04-23]
Dostupné z: `https://git-scm.com/`

[9] *Slack - Collaboration hub for work* [online] [2019-04-23]
Dostupné z: `https://slack.com/`

[10] *Článek Observable návrhový vzor* [online] [2019-04-23]
Dostupné z `https://en.wikipedia.org/wiki/Observer_pattern`

[11] *Jenkins - Build great things at any scale* [online] [2019-04-23]
Dostupné z: https://jenkins.io/

[12] *Dokumentace Jira Attlasian* [online] [2019-04-23]
Dostupné z `https://www.atlassian.com/software/jira`

[13] *Dokumentace Grunt - JavaScript Task Runner* [online] [2019-04-23]
Dostupné z `https://gruntjs.com/`

[14] *Dokumentace Less* [online] [2019-04-23]
Dostupné z `http://lesscss.org/`

[15] *Dokumentace Css* [online] [2019-04-23]
Dostupné z `https://developer.mozilla.org/en-US/docs/Web/CSS`

[16] *Dokumentace direktivy AngularJS - ngTransclude* [online] [2019-04-23]
Dostupné z `https://docs.angularjs.org/api/ng/directive/ngTransclude`

[17] *Dokumentace direktivy AngularJS - ngIf* [online] [2019-04-23]
Dostupné z `https://docs.angularjs.org/api/ng/directive/ngIf`

[18] *Dokumentace direktivy AngularJS - ngRepeat* [online] [2019-04-23]
Dostupné z `https://docs.angularjs.org/api/ng/directive/ngRepeat`

[19] *Dokumentace direktivy AngularJS - ngShow* [online] [2019-04-23]
Dostupné z `https://docs.angularjs.org/api/ng/directive/ngShow`

[20] *Dokumentace direktivy AngularJS - ngHide* [online] [2019-04-23]
Dostupné z `https://docs.angularjs.org/api/ng/directive/ngHide`

[21] *Článek RESTful API* [online] [2019-04-23]
Dostupné z `https://en.wikipedia.org/wiki/Representational_state_transfer`

[22] *Dokumentace MariaDB DB server* [online] [2019-04-23]
Dostupné z `https://mariadb.org/`

[23] *Dokumentace AngularJS testování* [online] [2019-04-23]
Dostupné z `https://docs.angularjs.org/guide/e2e-testing`

[24] *Článek Unix Timestamp* [online] [2019-04-23]
Dostupné z `https://en.wikipedia.org/wiki/Unix_time`